US 20050132210A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2005/0132210 A1

Yunus et al. (43) Pub. Date: Jun. 16, 2005

(54) HYBRID JAVA-C NETWORK APPLIANCE

(76) Inventors: **Mamoon Yunus**, Newton, MA (US); **Rizwan Mallal**, Waltham, MA (US); **Thomas C. Stickle**, St. James, NY (US)

Correspondence Address:
MORRISS O'BRYANT COMPAGNI, P.C.
136 SOUTH MAIN STREET
SUITE 700
SALT LAKE CITY, UT 84101 (US)

(21) Appl. No.: **10/909,848**

(22) Filed: **Aug. 2, 2004**

Related U.S. Application Data

(60) Provisional application No. 60/492,174, filed on Aug. 1, 2003.

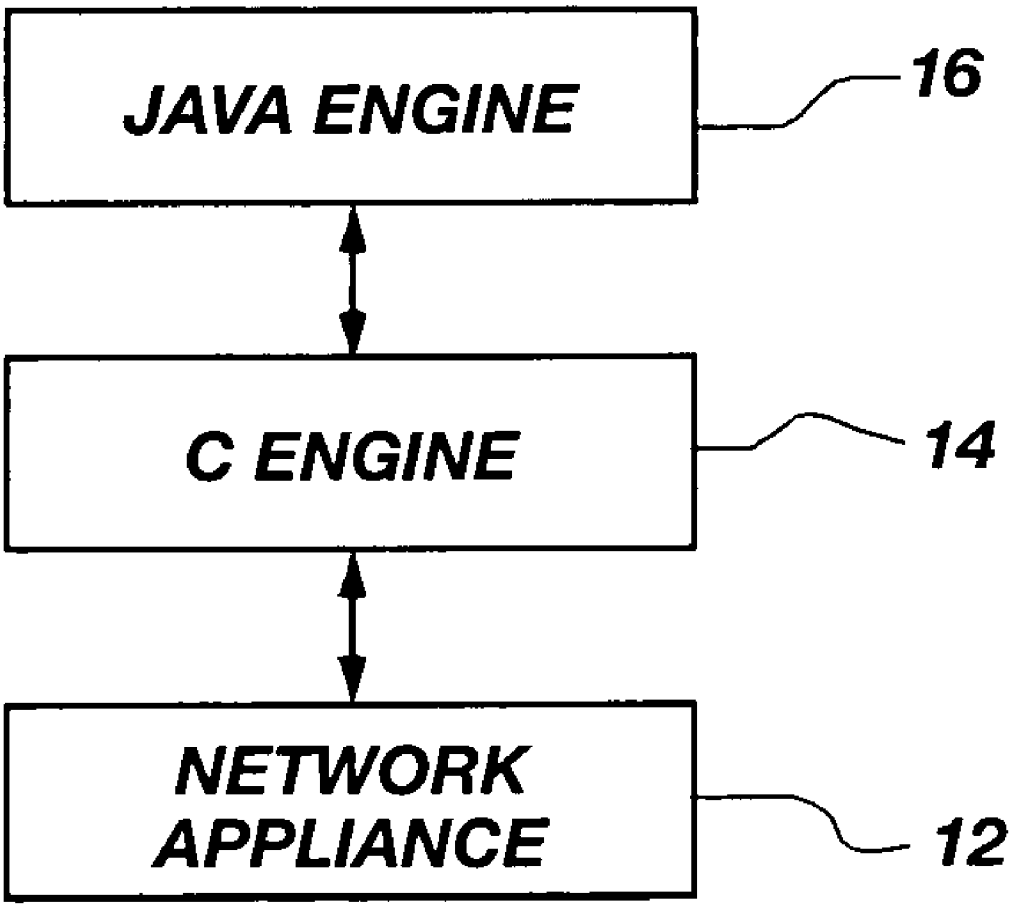
Publication Classification

(51) **Int. Cl.**[7] **G06F 11/30**

(52) **U.S. Cl.** **713/189**

(57) **ABSTRACT**

A network appliance that runs both C and Java integrated software to provide a flexible architecture for rapid prototyping of XML security functionality, including SSL acceleration, XML encryption, XML decryption, XML signature, and XML verification, while the network appliance continues to provide high-speed performance.



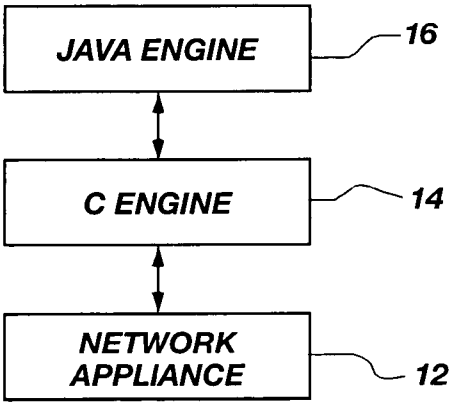

JAVA ENGINE
16
C ENGINE
14
NETWORK APPLIANCE
12

FIG. 1

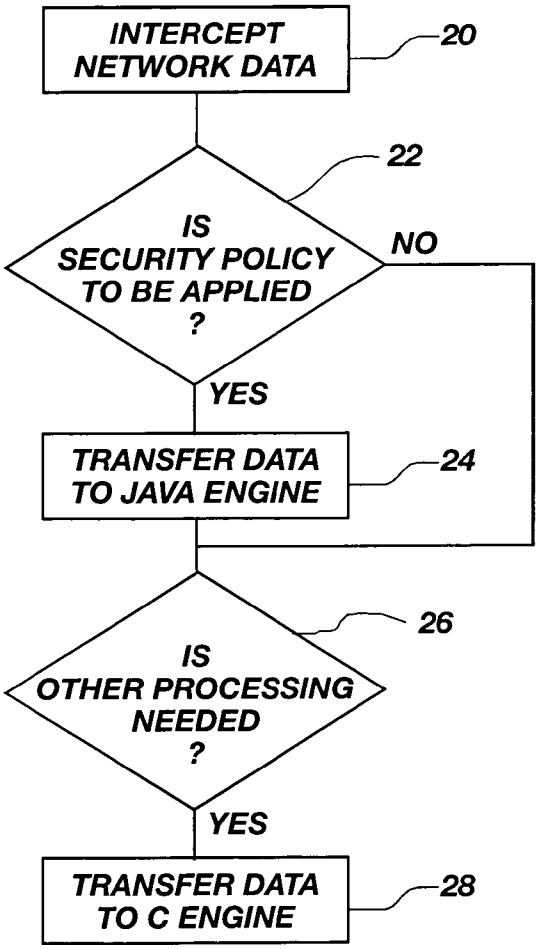

INTERCEPT NETWORK DATA
20
22
IS SECURITY POLICY TO BE APPLIED ?
NO
YES
TRANSFER DATA TO JAVA ENGINE
24
26
IS OTHER PROCESSING NEEDED ?
YES
TRANSFER DATA TO C ENGINE
28

FIG. 2

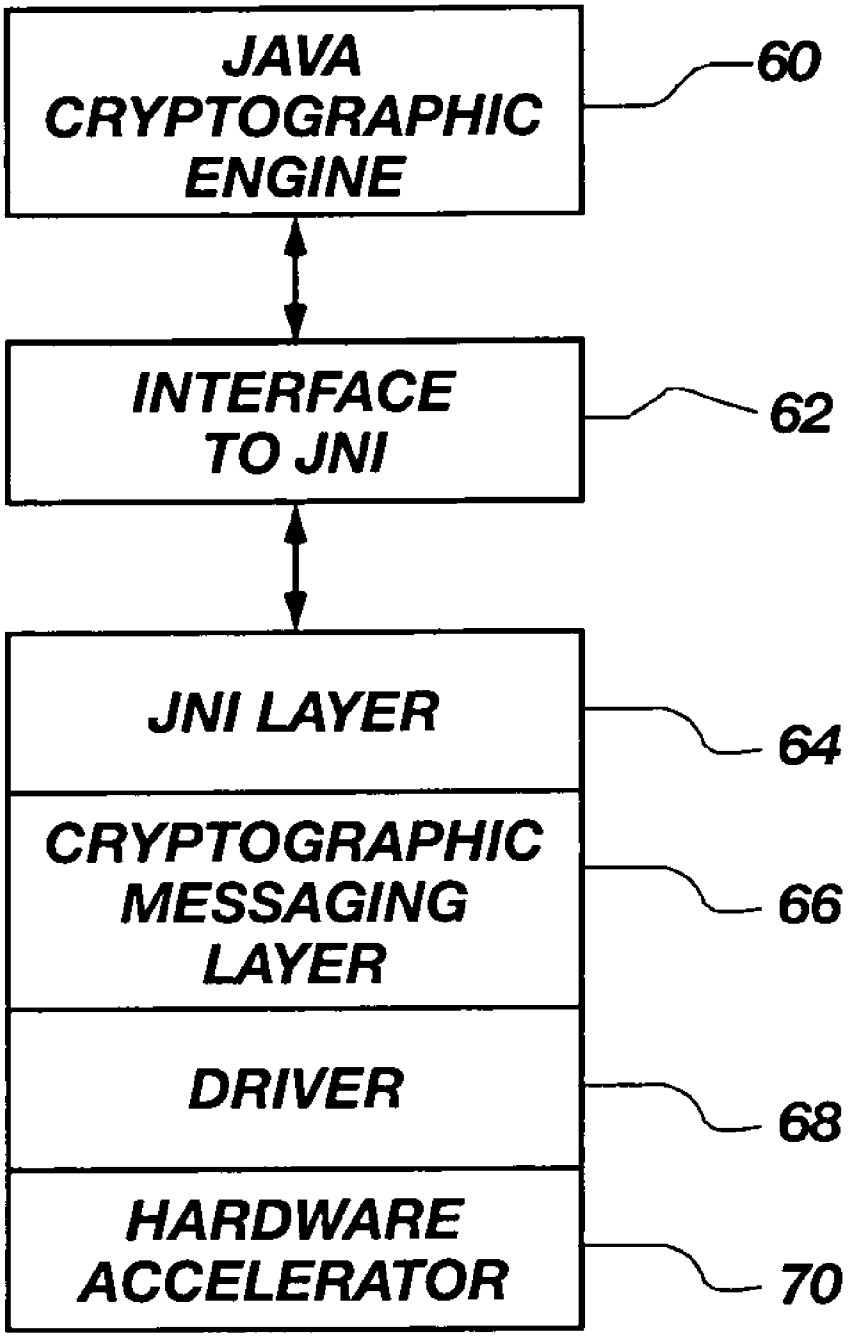

JAVA CRYPTOGRAPHIC ENGINE
60
INTERFACE TO JNI
62
JNI LAYER
64
CRYPTOGRAPHIC MESSAGING LAYER
66
DRIVER
68
HARDWARE ACCELERATOR
70

FIG. 3

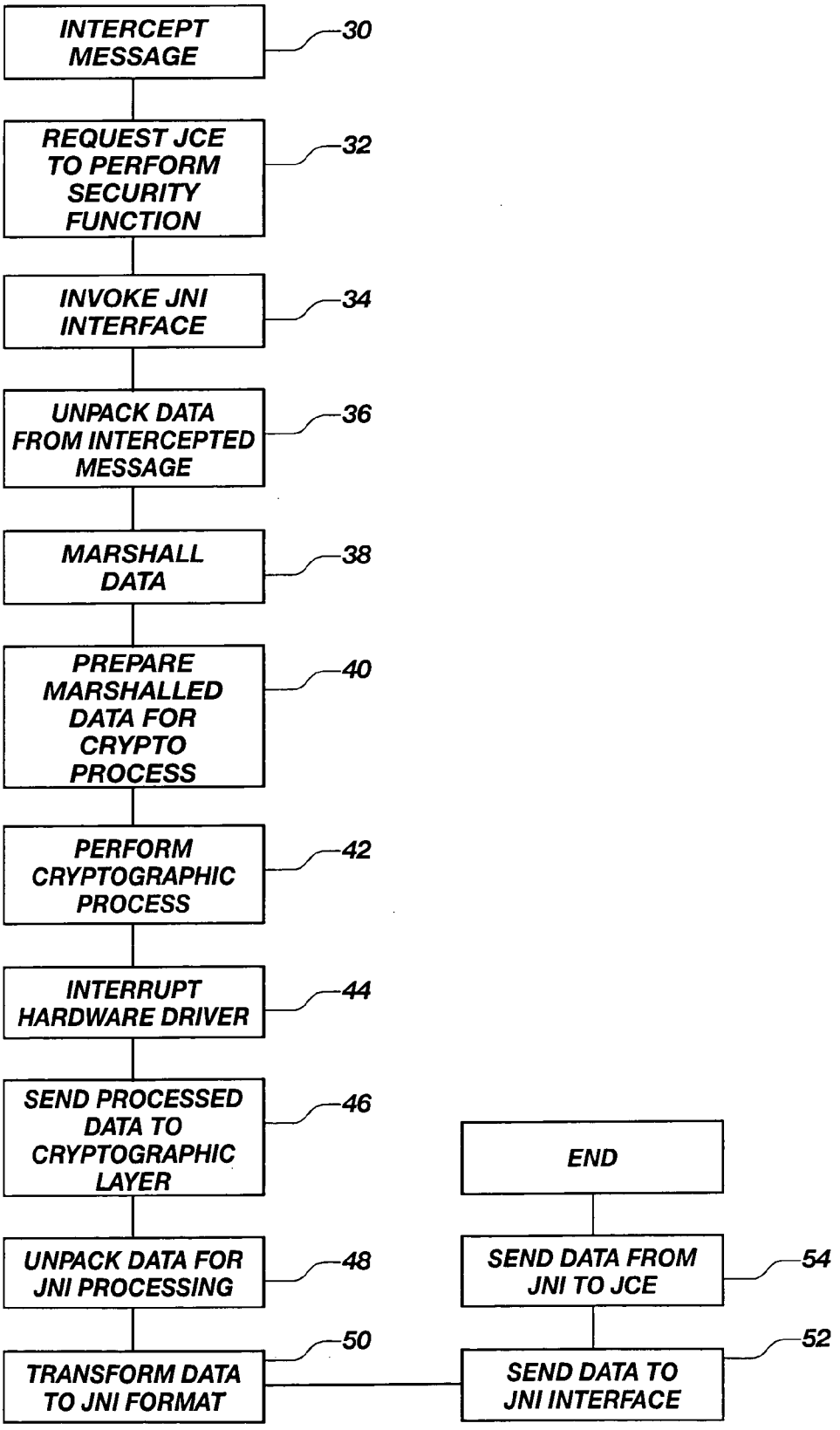

INTERCEPT MESSAGE
30
REQUEST JCE TO PERFORM SECURITY FUNCTION
32
INVOKE JNI INTERFACE
34
UNPACK DATA FROM INTERCEPTED MESSAGE
36
MARSHALL DATA
38
PREPARE MARSHALLED DATA FOR CRYPTO PROCESS
40
PERFORM CRYPTOGRAPHIC PROCESS
42
INTERRUPT HARDWARE DRIVER
44
SEND PROCESSED DATA TO CRYPTOGRAPHIC LAYER
46
UNPACK DATA FOR JNI PROCESSING
48
TRANSFORM DATA TO JNI FORMAT
50
END
SEND DATA FROM JNI TO JCE
54
SEND DATA TO JNI INTERFACE
52

FIG. 4

HYBRID JAVA-C NETWORK APPLIANCE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to and incorporates by reference provisional patent application Ser. No. 60/492,174, filed Aug. 1, 2003.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] This invention relates generally to a hardware network appliance running C and Java software and providing XML security functionality, wherein the C and Java code are both integrated in the appliance to provide SSL acceleration, XML encryption, XML decryption, XML signature, and XML verification, and wherein the network appliance can be deployed in different network topologies using the C and Java integrated code to thereby obtain improved speed and flexibility.

[0004] 2. Description of Related Art

[0005] The state of the art in XML security products is characterized by a network appliance that is designed to perform desired security functions. For example, SSL acceleration, XML encryption, XML decryption, XML signature, and XML verification can all be performed by these network appliances on network traffic that is intercepted.

[0006] Network traffic is intercepted by placing the network appliance in an in-line mode to intercept traffic that is targeted towards application or integration servers, or functioning as a proxy server in order to intercept all server requests.

[0007] Generally, a network appliance will run C code in its systems in order to provide the desired network functions. However, the C programming language can be cumbersome to use because it is not optimized for addressing network issues. Coding can also be a time-consuming process. Thus, it becomes more difficult to provide updated software for the network appliance that will enable the appliance to keep pace with rapid advancements in network functions.

[0008] Accordingly, what is needed is a network appliance that has a flexible architecture that will allow such things as cutting edge XML security features to be rapidly prototyped and deployed in the network appliance.

BRIEF SUMMARY OF THE INVENTION

[0009] It is an object of the present invention to provide a network appliance that can be rapidly updated with cutting edge security functionality.

[0010] It is another object to provide a network appliance having a flexible architecture that enables rapid prototyping.

[0011] It is another object to provide a network appliance that can be programmed using a language that is better suited for rapid prototyping and network functions.

[0012] It is another object to provide a network appliance that can be deployed in different network topologies.

[0013] It is another object to provide a network appliance that enables rapid prototyping without sacrificing high-speed performance.

[0014] In a preferred embodiment, the present invention is a network appliance that runs both C and Java integrated software to provide a flexible architecture for rapid prototyping of XML security functionality, including SSL acceleration, XML encryption, XML decryption, XML signature, and XML to provide high-speed performance.

[0015] These and other objects, features, advantages and alternative aspects of the present invention will become apparent to those skilled in the art from a consideration of the following detailed description taken in combination with the accompanying drawings.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0016] **FIG. 1** is a block diagram that illustrates the components of the system as made in accordance with the principles of the present invention.

[0017] **FIG. 2** is a flowchart of one embodiment of the present invention.

[0018] **FIG. 3** is a block diagram of functional layers of operations that are arranged in accordance with other the principles of the present invention.

[0019] **FIG. 4** is a flowchart of the steps that can be performed in another embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0020] Reference will now be made to the drawings in which the various elements of the present invention will be given numerical designations and in which the invention will be discussed so as to enable one skilled in the art to make and use the invention. It is to be understood that the following description is only exemplary of the principles of the present invention, and should not be viewed as narrowing the claims which follow.

[0021] The presently preferred embodiment of the invention is a system whereby all security functions can be accelerated through the careful use of security policies that enable the present invention to accelerate some security functions of a network. For example, consider the general flow of network data through the hardware shown in **FIG. 1** and shown as a flowchart in **FIG. 2**. **FIG. 1** shows a network appliance **12**. This network appliance **12** can be configured in an in-line mode wherein network traffic must pass through it to get to another side, or in a proxy mode. The goal of the present invention is to increase throughput of network traffic through the network appliance, especially when the network traffic must have a security process performed before the network traffic is permitted to continue on to its intended destination.

[0022] Accordingly in step **1 (20)**, network traffic will be intercepted by the network appliance **12**. As a TCP/IP packet enters one of the Ethernet ports of the network appliance **12**, a network driver of the present invention performs an analysis. The purpose of the analysis is to match the IP packet to a particular server policy in step **2 (22)**.

[0023] Further explanation is useful at this step. The TCP/IP packet is being transferred using TCP/IP protocols. TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data

and also guarantees that packets will be delivered in the same order in which they were sent. In contrast, IP specifies the format of packets. IP is thus combined with the higher level protocol TCP. IP enables a packet to have an address, but there is no direct link between the sender and the recipient. Thus, the present invention is simply analyzing the packet itself in step **2 (22)** to determine what security policy may need to be applied.

[0024] An example of a security policy is to direct all security functions to a Java engine **16** for processing in step **3 (24)**. Thus, if the IP packet is encrypted, the security function to be performed is decryption so that the intended recipient of the message can read the message. This security function must be considered only to be an example, and not limiting in any way. Other security functions that may be performed may be considered to include, but not be limited to, encryption, decryption, verification, and signing functions.

[0025] Accordingly, if the IP packet is analyzed and determined to match a policy that requires action on the part of the Java engine **16**, the IP packet is transferred from the network appliance **12** to the Java engine. The Java engine **16** processes a data portion of the IP packet. The next step **4 (26)** is to determine if more processing is required that is not performed by the Java engine **16**. If more processing is required, the next step **5 (28)** is to send the data to a C engine **14** for further processing. Further processing may include acceleration of security functions as explained and incorporated by reference from co-pending patent applications entitled JAVA CRYPTOGRAPHIC ENGINE TO CRTYPTO ACCELERATION INTEGRATION, and ARBITRARY JAVA LOGIC DEPLOYED TRANSPARENTLY IN A NETWORK, both filed on Aug. 1, 2003. After further processing of the data, the C engine **12** sends the data back to the Java engine **16**. Finally, the Java engine **16** sends the data in the form of an IP packet back to a network port.

[0026] It is important to recognize that the processes begin performed by the C engine **12** may be able to be implemented in hardware. For example, consider the process of cryptographic acceleration, which may include encryption, decryption, verification and signing processes. Cryptographic acceleration can be performed by the C engine **12**, but it can also be offloaded to dedicated hardware.

[0027] In this example of the present invention as shown in **FIGS. 3 and 4**, the security function that is to be performed is a public key operation as understood by those skilled in the art. Specifically, a public key may be known by everyone, and a private key is known only to the recipient of the message. The public key will be assumed to have been used to encrypt a message that has been intercepted in step **1 (30)** by the network appliance. It is assumed that it is known that the message contains a message that requires a cryptographic process to be performed.

[0028] After interception, a request enters the JCE layer **12** for a public key operation to be performed on data using, for example, the RSA cryptographic algorithm in step **2 (32)**. The next step **3 (34)** is for Java Native Interface (JNI) hooks to be invoked through the Interface to JNI **64**. In the present invention, the JNI hooks are only inserted in particular sections of the JCE **60** when public key operations are to be offloaded. The JNI hooks provide an interface to a C interface library which is shown as the JNI layer **64**.

[0029] The next step **4 (36)** is for the JNI layer **64** to take the requests for public key operations from the layers above and unpack the data from the message so that the data can be manipulated by the C language.

[0030] After being unpacked, the data is marshaled by the Cryptographic Messaging layer **66** in step **5 (38)**. Data marshalling is required when passing the output parameters of a program written in one language as input to a program written in another language. In this case, the purpose of data marshalling is to gather data and transform it into a standard format. In order for an object to be moved around a network, it must be converted into a data stream that corresponds with the packet structure of the network transfer protocol.

[0031] After marshalling, the data is passed on to the Hardware Driver **68**. The purpose of the Hardware Driver **68** is to prepare Hardware Accelerator **70** to perform the desired operation in step **6 (40)**. In this case, the Hardware Accelerator **70** is being prepared to perform decryption of the intercepted message.

[0032] Once the RSA decryption operation is performed in step **7 (42)**, the Hardware Driver **68** is interrupted by the Hardware Accelerator **70** in step **8 (44)**. The Hardware Driver **68** passes the decrypted data back to the Cryptographic Messaging Layer **66** in step **9 (46)**. The Cryptographic Messaging Layer **66** unpacks the decrypted data in raw C format in step **10 (48)** for the JNI Layer **64** to transform to JNI format in step **11 (50)** which the upper Java layers will understand. Once the JNI Interface **62** hooked to the JCE **60** receives the result in JNI format in step **12 (52)**, the JNI Interface **62** unpacks the decrypted data and sends the results to the JCE **60** in step **13 (54)**.

[0033] It should be understood that the sequence above is followed for any security functions that can be offloaded to a hardware accelerator, and is not limited to the example of RSA decryption described above.

[0034] It is envisioned that the Java and C proxy engines will be ported to a software platform on a desktop PC or a notebook PC running, for example, Windows 2000 or Windows XP. However, this should not be considered a limiting factor, and the present invention can be ported to other operating systems and other hardware platforms as well.

[0035] The advantages of the present invention over the prior art are substantial. The present invention is versatile because of its platform independence that is enabled by the use of the Java language. The offloading is transparent to operation of the network appliance. Use of the Java language inherently means that the prototyping of changes and improvements is rapid because of the ease of use of the Java language. Performance of the offloaded security functions is substantially increased because of hardware that is dedicated to the desired security functions, instead of using a general purpose CPU. Furthermore, high speed performance is maintained because of the use of the Java language.

[0036] It is to be understood that the above-described arrangements are only illustrative of the application of the principles of the present invention. Numerous modifications and alternative arrangements may be devised by those skilled in the art without departing from the spirit and scope of the present invention. The appended claims are intended to cover such modifications and arrangements.

What is claimed is:

1. A method for applying security policies to data in a network, said method comprising the steps of:

1) intercepting data being transferred across the network;

2) determining that a policy to be performed can be offloaded for acceleration;

3) utilizing a Java engine to offload the data; and

4) performing a desired security function.

2. The method as defined in claim 1 wherein the method further comprises the step of utilizing a proxy engine to transparently offload the data.

3. The method as defined in claim 2 wherein the method further comprises the step of performing the security function in software.

4. The method as defined in claim 3 wherein the method further comprises the step of performing the security function in a C language module.

5. The method as defined in claim 4 wherein the method further comprises the step of performing the security function in dedicated hardware.

6. A system for applying security policies to data in a network, said system comprised of:

a C engine for performing offloaded functions; and

a Java engine for offloading a message to the C engine for processing.

7. The system as defined in claim 6 wherein the system is further comprised of a hardware accelerator for assisting the C engine in performing security functions.

8. A method for accelerating processing of security functions in a network, said method comprising the steps of:

1) intercepting data being transferred across the network;

2) determining that a security function to be performed can be offloaded for hardware acceleration;

3) utilizing a proxy engine to transparently offload the data; and

4) performing the security function in hardware.

9. The method as defined in claim 8 wherein the method of utilizing a proxy engine further comprises the step of utilizing a Java Cryptographic Engine (JCE) to transparently offload the data to be offloaded.

10. The method as defined in claim 9 wherein the method further comprises the step of entering a request in the JCE layer for a cryptographic function to be performed.

11. The method as defined in claim 10 wherein the method further comprises the step of invoking Java Native Interface (JNI) hooks to function as an interface to an operating system specific programming language interface library.

12. The method as defined in claim 11 wherein the method further comprises the step of invoking Java Native Interface (JNI) hooks in a JNI layer to function as an interface to a C programming language interface library.

13. The method as defined in claim 12 wherein the method further comprises the step of unpacking data from the intercepted message so that the data can be manipulated in the operating system specific programming language.

14. The method as defined in claim 13 wherein the method further comprises the step of marshalling the data so that the data can be transformed to a standard format.

15. The method as defined in claim 14 wherein the method further comprises the step of marshalling the data in a cryptographic messaging layer.

16. The method as defined in claim 15 wherein the method further comprises the step of marshalling the data so that the data can be transferred across a network having a specific network packet protocol.

17. The method as defined in claim 16 wherein the method further comprises the step of transferring the data to a hardware driver.

18. The method as defined in claim 17 wherein the method further comprises the step of using the hardware driver to prepare a hardware accelerator for receiving data to be cryptographically processed.

19. The method as defined in claim 18 wherein the method further comprises the step of transferring the data from the hardware driver to the hardware accelerator for cryptographic processing.

20. The method as defined in claim 19 wherein the method further comprises the step of selecting the type of cryptographic processing to be performed by the hardware accelerator from the group of cryptographic processes comprised of encrypting, decrypting, verification and signing.

21. The method as defined in claim 20 wherein the method further comprises the step of interrupting the hardware driver when the hardware accelerator has completed its cryptographic processing of the data so that the data can be transferred to a next destination.

22. The method as defined in claim 21 wherein the method further comprises the step of transferring the data from the hardware driver to the cryptographic messaging layer.

23. The method as defined in claim 22 wherein the method further comprises the step of unpacking the data.

24. The method as defined in claim 23 wherein the method further comprises the step of transferring the data from the cryptographic messaging layer to the JNI layer.

25. The method as defined in claim 24 wherein the method further comprises the step of transforming the data in the JNI layer.

26. The method as defined in claim 25 wherein the method further comprises the step of transferring the data from the JNI layer to the JCE layer through the JNI interface.

* * * * *